04-24-00

A

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Inventorship.....Perlin et al.
Applicant..... Microsoft Corporation
Attorney's Docket No. MS1-385US
Title: Interlaced Protocol for Smart Card Application Development

TRANSMITTAL LETTER AND CERTIFICATE OF MAILING

To: Commissioner of Patents and Trademarks,
Washington, D.C. 20231

From: Michael A. Proksch (Tel. 509-324-9256; Fax 509-323-8979)
Lee & Hayes, PLLC
421 W. Riverside Avenue, Suite 500
Spokane, WA 99201

The following enumerated items accompany this transmittal letter and are being submitted for the matter identified in the above caption.

1. Specification–title page, plus 40 pages, including 49 claims, Abstract and Appendix A
2. Transmittal letter including Certificate of Express Mailing
3. 7 Sheets Formal Drawings (Figs. 1-8)
4. Return Post Card

Large Entity Status [x] Small Entity Status [ ]

Date: 4/19/2000 By: Michael Proksch
Michael A. Proksch
Reg. No. 43,021

CERTIFICATE OF MAILING

I hereby certify that the items listed above as enclosed are being deposited with the U.S. Postal Service as either first class mail, or Express Mail if the blank for Express Mail No. is completed below, in an envelope addressed to The Commissioner of Patents and Trademarks, Washington, D.C. 20231, on the below-indicated date. Any Express Mail No. has also been marked on the listed items.

Express Mail No. (if applicable) EL580803890

Date: 4.19.2000 By: Lori A. Vierra
Lori A. Vierra

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

INTERLACED PROTOCOL FOR SMART CARD APPLICATION DEVELOPMENT

Inventor(s):
Eric C. Perlin
Vinay Deo
David Milstein
Gilad Odinak
Scott B. Guthery
Klaus U. Schutz

ATTORNEY'S DOCKET NO. MS1-385US

PRIORITY INFORMATION

This application expressly claims the benefit of the filing date established with U.S. Provisional Application No. 60/133,624 filed May 11, 1999 by Perlin, et al. and commonly assigned to the assignee of the present application.

TECHNICAL FIELD

This invention relates to a class of devices commonly known as smart cards and, in particular, to an interlaced protocol for smart card application development.

BACKGROUND OF THE INVENTION

Today there is increasing use of integrated circuit (IC) cards, colloquially referred to as "smart cards", in place of, or in addition to, conventional magnetic stripe cards ("mag cards"). A smart card is a thin card embedded with a memory device (volatile and/or non-volatile) and associated programmable or non-programmable logic. Unlike the mag card that merely stores "static" information (e.g., a credit card account number), a smart card can add, delete and otherwise manipulate information stored on the card. Accordingly, smart cards are capable of storing and executing applications to carry out one or more functions within a smart card.

While the physical dimensions and processing features of the smart card give rise to potentially limitless applications, the reality is that smart card applications are typically only developed for large scale markets, e.g., banking, security and transportation applications. One reason for this limited growth lies in the cost associated with smart card application development. There are several reasons why smart card application development is a costly undertaking, not the

least of which is the "closed" nature of the smart card and the limited processing, memory and input/output resources of the smart card.

A smart card is often referred to as a "closed" system because, for security purposes, a smart card is purposefully designed to not expose its memory, intermediate system states or data and address bus information to external devices. To do so would render it susceptible to unauthorized access (hacking) and fraud. While its closed nature is useful for secure applications such as banking transactions, it makes it difficult to utilize prior art smart cards for development purposes. It is to be appreciated that application development often requires access to memory or bus values, or system state information during intermediate processing steps, access that has been specifically designed out of the smart card.

Another encumbrance to the smart card application designer is the limited resources of the smart card. That is, due to the physical and processing constraints placed on the smart card, prior art smart cards do not enjoy any dedicated debug facilities. Aside from the limited processing and memory attributes of a smart card, a smart card typically has but a single, bi-directional input/output (I/O) port. The communication bandwidth of this single I/O port is typically consumed to support execution of the smart card application itself, leaving little to no communication bandwidth to support debug features. Thus, application development using a smart card itself is virtually impossible. Consequently the development of applications for a smart card currently requires the use of an in-circuit emulator (ICE) and an associated, often proprietary software development application.

A ICE system is typically comprised of a printed circuit card coupled to a computer system executing a proprietary software development application

associated with the printed circuit card (emulator). The printed circuit card is designed to emulate the functionality of the smart card, while providing additional debug facilities (e.g., I/O ports, memory buffers, address and data lines and the like), thereby providing the developer with the necessary access to adequately debug their applications in development. One limitation of such smart card development systems is that the ICE and proprietary development application are chip-specific. Thus, an emulator for smart card employing a Siemens processor will not work with an emulator employing a Philips or Motorola processor without significant hardware modification. Moreover, the software development application executing on the computer system is also chip-specific, with an associated chip-specific compiler, linker and debugger, and often require that a developer learn the "programming language" of the development tool. Consequently, an application developed on one ICE system cannot be utilized (or directly ported to) a smart card employing a different processor without costly modification.

As a result of each of the foregoing limitations, smart card application development is a costly undertaking, typically performed by the large corporations that stand to profit from the sale of millions of smart cards. History has shown that in order for a new technology to blossom, "grass roots" application development is required. That is, a technology will not truly become a pervasive technology unless and until it is infused with the vitality and creativity of individual programmers and small development companies.

Thus, an improved application development environment is required for smart card applications that is unencumbered by the limitations commonly associated with the prior art. One such solution is presented below.

SUMMARY OF THE INVENTION

This invention concerns an integrated circuit (IC) card, such as a smart card and, more particularly, an interlaced protocol for smart card application development.

In accordance with a first aspect of the invention, an IC is presented comprising an input/output (I/O) interface and a smart card development interface (SCDI), coupled to the I/O interface, to receive and identify debug frames interlaced within a normal communication flow between the smart card and an application executing on a host system.

According to another aspect of the invention, a computer system is presented comprising a client development interface (CDI), to receive and identify debug frames interlaced within a normal communication flow received from a communicatively coupled integrated circuit (IC) card.

According to yet another aspect of the present invention, a protocol enabling smart card application debugging using an IC card is introduced. In particular, a protocol facilitating communication between a host system and an IC card is presented comprising a plurality of application frames and one or more debug frames. The application frames facilitate communication between a host application and one or more smart card resources. The debug frame is interlaced with the application frames comprising normal communication flow to invoke one or more smart card resources. In one embodiment, the host application and the debug application are executing on separate host systems utilizing the resources of an IC card.

It is to be appreciated that combination of the foregoing aspects of the present invention gives rise to an innovative smart card application development system comprising a computer system endowed with the client development interface (CDI), and a smart card incorporating the smart card development interface, wherein communication between the IC card and the computer system adheres to a transport protocol supporting Application Protocol Data Units (APDU) (i.e., a normal communication flow) and innovative Debug Protocol Data Units (DPDU). Accordingly, it will be appreciated that the innovative development interfaces, i.e., the CDI and the SCDI, respectively, represent a significant advancement in smart card application development, enabling a developer to develop smart card applications using an actual IC card, rather than the costly emulators required in the prior art.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of an example smart card application development system including a computer system and a smart card;

Fig. 2 is a block diagram of an example computer system including a client development interface, suitable for use in the smart card application development system of Fig. 1;

Fig. 3 is a block diagram of an example client development interface including a debug filter, according to one aspect of the present invention;

Fig. 4 is a block diagram of an example smart card including a smart card development interface, suitable for use in the smart card application development system of Fig. 1;

Fig. 5 is a block diagram of an example smart card development interface including a debug filter, according to one aspect of the present invention;

Fig. 6 graphically illustrates an example communication flow including a diagram of an example debug frame suitable for use in the application development system of Fig. 1;

Fig. 7 is a flow chart of an example method for debugging an IC card application using an interlaced debug protocol, according to one aspect of the present invention; and

Fig. 8 illustrates a signaling diagram for an example communication flow between a host system and a smart card utilizing the interlaced debug protocol of the present invention.

DETAILED DESCRIPTION

Example Development System

Fig. 1 illustrates an application development system 100 comprising a computer system 102 coupled to smart card 104 in which a user can develop and debug application code for use on a smart card such as, but not limited to, smart card 104. As shown, the development system of Fig. 1 depicts computer system 102 coupled to smart card 104 via a card reader 106 and a communication medium 108. The communication medium 108 is intended to represent any of a number of typical communication links including, but not limited to, a proprietary data bus, an industry standard data bus, a local area network (LAN), a wide area network (WAN), or a global area network (e.g., the Internet). In this regard, as will be developed more fully below, the innovative application development system 100 facilitates development of smart card applications using an actual smart card 104

coupled to one or more computer system(s) (e.g., 102) via a communications network 108 and a card reader 106.

Smart card 104 comprises an innovative smart card development interface (SCDI) 110 coupled to an operating system (OS) 112. According to one aspect of the invention, to be developed more fully below, SCDI 110 receives and identifies debug frames interlaced with "normal" application frames within the normal communication flow between the smart card 104 and an application executing within an application development tool running on computer system 102.

It is to be appreciated that, but for the innovative smart card development interface 110, smart card 104 and its operating system 112 with associated system resources is intended to represent any of a broad range of integrated circuit cards and their operating systems commonly known in the art. That is, any smart card endowed with the innovative smart card development interface 110 is suitable for use within the innovative smart card application development system 100 of Fig. 1.

It is noted that, in addition to the illustrated smart cards, the IC device might be embodied in other forms, such as an electronic wallet, a personal digital assistant, a smart diskette (i.e., an IC-based device having a form factor and memory drive interface to enable insertion into a floppy disk drive), a PC card (formerly PCMCIA card), and the like. Generally, the integrated circuit card 104 is characterized as an electronic device with limited processing capabilities and memory wherein large size number crunching is impractical.

Card reader 106 provides a necessary interface between smart card reader 104 and a computing system such as, e.g., computer system 102. Card readers are typically designed to support any of a number of standardized communication

protocols supported within the smart card community and, in this way, can typically accommodate smart cards adhering to any of the recognized communication standards from any smart card manufacturer. In this regard, card reader 106 is not chip- or card-specific. For purposes of this discussion, card reader 106 includes the necessary hardware and software resources required to support the interlaced debug protocol of the present invention. Consequently, card reader 106 is merely intended to be illustrative of card readers typically known within the art.

Computer system 102 is depicted within Fig. 1 as comprising an innovative client development interface (CDI) 114, a plurality of executable applications 116 including application development tool 118 with a debug environment, and an operating system 120, coupled as shown. The application development tool 118 enables a user to code and debug a smart card application, utilizing a debug environment that generates debug frames. According to one aspect of the present invention, the CDI 114 marshals and interlaces the debug frames with application frames (normal communication flow) generated by the application executing within development tool 118.

As will be developed in more detail, below, CDI 114 receives debug frames from an external application, and interlaces debug frames within the normal communication flow between computer system 102 and smart card 104. According to an exemplary embodiment, the debug frames are generated by a debug environment within application development tool 118 and sent to CDI 114. CDI 114 includes a debug filter to identify debug frames. The debug frames are generated within a unique identifiable attribute such as, for example, embedding an invalid source and/or destination address (e.g., FF hex) within the debug frame.

In addition, CDI 114 receives and identifies debug frames, i.e., response debug frames, sent from smart card 104. The debug filter of CDI 114 identifies the debug frames (e.g., by the invalid source/destination address) and promotes the response to a debug environment, while normal application frames are promoted to the application executing within application development tool 118. But for the client development interface 114, computer system 102, applications 116 and operating system 120 are each intended to represent any of a number of commonly known computer systems, applications and operating systems, respectively, known in the art.

According to one innovative aspect of the present invention, development application 118 is intended to be any of a number of known software development applications (also referred to as software development "tools"). Examples of such software development tools include Visual Basic or Visual C/C++ from Microsoft Corporation of Redmond, WA. Thus, a smart card application is developed using computer 102 and a typical software development tool 118, utilizing the interlaced debug protocol supported by CDI 114 and SCDI 110 to invoke and interrogate smart card resources to verify the integrity of the developed code. By using common software development tools such as those described above, the smart card application development system 100 does not require the chip-specific, often proprietary software development application and associated compilers, linkers and debuggers that are typical of the cumbersome prior art development systems.

As alluded to above, the inclusion of the innovative CDI 114 and SCDI 110 within development system 100 support an interlaced debug protocol that interlaces debug frames with standard application frames comprising a normal communication flow between computer system 102 and smart card 104.

According to one embodiment, the debug frames are generated in response to user interaction with a debug environment of application development tool 118 executing a smart card application. The debug frames are sent to CDI 114, which identifies the debug frames and interlaces such frames with the normal application frames (generated by the application executing within the application development tool) and sent to smart card 104 via card reader 106 and communication medium 108. SCDI 110 receives the communication from computer system 102, identifies and routes the debug frames to a debug monitor, while application frames are promoted to an appropriate application/resource of the smart card (i.e., as identified by a source/destination address). The received debug frames include debug instructions which selectively invoke smart card resources (e.g., API's, device drivers, applications, etc.), providing a user with a heretofore unavailable view of system state information while an application is executing on the smart card. As described above, this state information is priceless during application development.

It should be appreciated that the interlaced debug protocol supported by the innovative CDI 114 and SCDI 110 of the present invention enables a user to employ an otherwise ordinary development tool 118 on an otherwise ordinary computer system to directly utilize the resources of smart card 104 to code and debug smart card applications. In addition to cost and ease of use advantages over the prior art, development system 100 represents a significant improvement over prior art development systems in that the applications developed using the present invention are easily ported from one smart card to another (i.e., the application is not chip specific, as is the case of systems developed using an ICE system). Moreover, insofar as the actual smart card resources are utilized during the

development, there is less of a chance for hidden bugs or other undetected compatibility problems as measured against prior art development systems. In this regard, application development system 100 represents a significant improvement over the prior art in terms of cost, ease of use and quality of the end product.

Although the exemplary embodiment above discusses application development and debugging using application development tool 118 and an integrated debug environment, this is for ease of explanation only. An alternate implementation may well provide an independent debug monitor executing on computer 102, in communication with CDI 114 to interlace debug frames within the normal communication flow generated by a host application, independently executing on computer 102. Similarly, a debug application may well be embedded within SCDI 110, or executing on smart card 104 as an independent entity. Thus, the description above and below is to be regarded as merely illustrative, and not limiting, of the spirit and scope of the present invention.

Example Computer System

In the discussion herein, the invention is described in the general context of computer-executable instructions, such as program modules, being executed by one or more conventional computers. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, personal digital assistants, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and

the like. In a distributed computer environment, program modules may be located in both local and remote memory storage devices.

Fig. 2 shows a general example of a computer system 102 incorporating the teachings of one aspect of the present invention, and suitable for use within the smart card application development system 100. It will be evident, from the discussion to follow, that computer 102 is intended to represent any of a class of general or special purpose computing platforms which, when endowed with the innovative client development interface 114, is suitable for use in smart card application development system 100. In this regard, the following description of computer system 102 is intended to be merely illustrative, as computer systems of greater or lesser capability may well be substituted without deviating from the spirit and scope of the present invention.

As shown, computer 102 includes one or more processors or processing units 132, a system memory 134, and a bus 136 that couples various system components including the system memory 134 to processors 132.

The bus 136 represents one or more of any of several types of bus structures, including a memory bus or memory controller, a peripheral bus, an accelerated graphics port, and a processor or local bus using any of a variety of bus architectures. The system memory includes read only memory (ROM) 138 and random access memory (RAM) 140. A basic input/output system (BIOS) 142, containing the basic routines that help to transfer information between elements within computer 102, such as during start-up, is stored in ROM 138. Computer 102 further includes a hard disk drive 144 for reading from and writing to a hard disk, not shown, a magnetic disk drive 146 for reading from and writing to a removable magnetic disk 148, and an optical disk drive 150 for reading from or

writing to a removable optical disk 152 such as a CD ROM, DVD ROM or other such optical media. The hard disk drive 144, magnetic disk drive 146, and optical disk drive 150 are connected to the bus 136 by a SCSI interface 154 or some other suitable bus interface. The drives and their associated computer-readable media provide nonvolatile storage of computer readable instructions, data structures, program modules and other data for computer 102. Although the exemplary environment described herein employs a hard disk 144, a removable magnetic disk 148 and a removable optical disk 152, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, random access memories (RAMs) read only memories (ROM), and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk 144, magnetic disk 148, optical disk 152, ROM 138, or RAM 140, including an operating system 158, one or more application programs 160 including, for example, the innovative client development interface 114 or application development tool 118, other program modules 162, and program data 164. A user may enter commands and information into computer 102 through input devices such as keyboard 166 and pointing device 168. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are connected to the processing unit 132 through an interface 170 that is coupled to bus 136. A monitor 172 or other type of display device is also connected to the bus 136 via an interface, such as a video adapter 174. In addition to the monitor 172, personal computers often include other peripheral output devices (not shown) such as speakers and printers.

As shown, computer 102 operates in a networked environment using logical connections to one or more remote computers, such as a remote computer 176. The remote computer 176 may be another personal computer, a personal digital assistant, a server, a router or other network device, a network "thin-client" PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computer 102, although only a memory storage device 178 has been illustrated in Fig. 2.

As shown, the logical connections depicted in Fig. 2 include a local area network (LAN) 180 and a wide area network (WAN) 182. Such networking environments are commonplace in offices, enterprise-wide computer networks, Intranets, and the Internet. In one embodiment, remote computer 176 executes an Internet Web browser program such as the "Internet Explorer" Web browser manufactured and distributed by Microsoft Corporation of Redmond, Washington to access and utilize online services.

When used in a LAN networking environment, computer 102 is connected to the local network 180 through a network interface or adapter 184. When used in a WAN networking environment, computer 102 typically includes a modem 186 or other means for establishing communications over the wide area network 182, such as the Internet. The modem 186, which may be internal or external, is connected to the bus 136 via a serial port interface 156. In a networked environment, program modules depicted relative to the personal computer 102, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Generally, the data processors of computer 102 are programmed by means of instructions stored at different times in the various computer-readable storage media of the computer. Programs and operating systems are typically distributed, for example, on floppy disks or CD-ROMs. From there, they are installed or loaded into the secondary memory of a computer. At execution, they are loaded at least partially into the computer's primary electronic memory. The invention described herein includes these and other various types of computer-readable storage media when such media contain instructions or programs for implementing the innovative steps described below in conjunction with a microprocessor or other data processor. The invention also includes the computer itself when programmed according to the methods and techniques described below. Furthermore, certain sub-components of the computer may be programmed to perform the functions and steps described below. The invention includes such sub-components when they are programmed as described. In addition, the invention described herein includes data structures, described below, as embodied on various types of memory media.

For purposes of illustration, programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of the computer, and are executed by the data processor(s) of the computer.

Fig. 3 illustrates a block diagram of an example client development interface (CDI) 114, suitable for use in computer system 102. As shown, CDI 114 comprises control logic 302, a debug filter 304 and memory 306, each coupled as depicted. CDI 114 may well be characterized as an abstraction layer, enabling

higher level applications and resources of computer 102 to access and utilize resources of smart card 104 endowed with SCDI 110 (which may also be characterized as an abstraction layer). With respect to the present invention, the features of CDI 114 are invoked upon detecting a debug frame by debug filter 304. Debug filter 304 detects debug frames by analyzing each of the frames received by the smart card for characteristics denoting a debug frame. According to one implementation, debug filter 304 detects an invalid source and/or destination address in a received frame marking the frame as a debug frame. It is to be appreciated, however, that debug filter 304 can be configured to detect alternative characteristics within a communication frame marking the frame as a debug frame.

Control logic 302 is intended to represent any of a broad range of logic known in the art. In one implementation, control logic 302 is a processor, while in alternate embodiments control logic is a microcontroller, a programmable logic array, or a series of executable instructions which perform logic functions. Control logic 302 communicates with smart card 104 in any of a plurality of standard smart card communication protocols. In alternate embodiments, non-standard protocols may well be used to communicate between controller 302 and smart card 104 such as, for example, a unique development communication protocol. Although not specifically denoted, it is to be appreciated that controller 302 communicates with smart card 104, and any other peripheral for that matter, via the communication resources of operating system 120. Insofar as such resources are well known in the art, they need not be described further here.

Upon detecting the arrival of a debug frame by debug filter 304, control logic 302 marshals the debug frame parameters and interlaces the completed debug frame with application frames comprising the normal communication flow

between computer 102 and smart card 104. According to one implementation, control logic 302 analyzes the task to be performed by the received debug frame, and places the debug frame at an appropriate point in the normal communication flow for transmission to smart card 104 via card reader 106 and communication medium 108.

Memory 306 includes one or more buffers wherein control logic 302 communication frames for transmission to smart card 104. In this regard, memory 306 is intended to represent any of a number of alternate memory devices commonly known to those in the art.

Although depicted as a separate functional element, those skilled in the art will appreciate that CDI 114 may well be integrated within and utilize the control features associated with the application development tool 118. In one implementation, for example, control logic 302 and memory 306 may well be supplied by a debug environment within the application development tool 118, wherein CDI 114 is comprised solely of debug filter 304. Accordingly, the teachings of the present invention may well be practiced with variation from the exemplary embodiment without deviating from the spirit and scope of the present invention.

Example Smart Card

Fig. 4 illustrates a block diagram of an example IC card 104 suitable for use within the application development system 100 of Fig. 1. In addition to the innovative smart card development interface (SCDI) 110 and an operating system with associated smart card resources 112, IC card 104 is shown comprising an input/output interface 402, memory 406 having stored therein a plurality of

executable applications and/or applets 404, and control logic 408. As discussed above, except for the inclusion of innovative smart card development interface (SCDI) 110, to be described more fully below, smart card 104 is intended to represent any of a broad category of integrated circuit (IC) cards commonly known in the art. Thus, but for SCDI 110, each of I/O 402, applets 404, memory 406 and control logic 408 are likewise commonly known within the art and, consequently, will not be further described here.

Fig. 5 illustrates a block diagram of an example SCDI 110, suitable for use within any IC card such as, e.g., smart card 104 of application development system 100 in Fig. 1. In accordance with the example implementation of Fig. 5, SCDI 110 is shown comprising control logic 502, debug filter 504, debug monitor 506 and memory 508, coupled as depicted. According to one implementation, SCDI 110 is invoked by CDI 114 upon receipt of a debug frame interlaced within the normal communication flow between computer 102 and smart card 104.

According to this exemplary implementation, control logic 502 may be any of a plurality of discrete logic and/or coded logic functions commonly known in the art such as, for example, a processor, a controller, or a plurality of executable instructions which implement such functionality. Similarly, memory 508 is intended to represent any of a number of memory devices known in the art.

As above, debug filter 504 identifies debug frames within the received communication flow by analyzing each of the frames received by the smart card for characteristics denoting a debug frame. According to one implementation, debug filter 504 detects an invalid source and/or destination address in a received frame marking the frame as a debug frame. It is to be appreciated, however, that

debug filter 504 can be configured to detect alternative characteristics within a communication frame marking the frame as a debug frame.

Debug monitor 506 selectively invokes one or more debugging features in response to receipt of a debugging frame, as identified by debug filter 504. According to one implementation, control logic 502 receives a debug frame, as detected by debug filter 504, and promotes the debug frame to debug monitor 506. Debug monitor controls and/or interrogates smart card resources (e.g., API's, device drivers, applications, etc.) in accordance with the debug instructions contained within the debug frame. As will be described more fully below, the debug monitor 506 can read/write smart card memory406, get/set breakpoints in a smart card application 404, sequentially step a smart card application 404, run a smart card application 404, release a smart card application 404, and obtain the context of control logic 408. Although depicted as an element of SCDI 110, debug monitor 506 may well be implemented as an independent debugging application resident on smart card 104, without deviating from the spirit and scope of the present invention.

Example Data Structures

Fig. 6 is a graphical illustration of a communication flow between a host computer 102 and smart card 104 including one or more interlaced debug frames, according to one aspect of the present invention. The innovative communication flow 600 is shown comprising a plurality of application protocol data units (APDU's) 602 and 606, selectively interlaced with one or more debug protocol data unit (DPDU) 604 which enables an independent debug application to share communication resources with a host application to control and interrogate the

otherwise closed architectural resources of a smart card. It is to be appreciated that, although depicted at the APDU level this is for ease of explanation only. That is, alternate embodiments are contemplated wherein interlacing of debug frames is performed at a block-level, i.e., at a lower level of communication. In this regard, it is to be appreciated that the DPDU is a construct of convenience to illustrate the features of the present invention. Alternate implementations at using lower-level or higher-level communication protocols are anticipated within the scope and spirit of the invention.

According to one implementation, the communication flow is a command/response communication protocol, initiated by the host system (e.g., computer 102). As alluded to above, the APDU's 602, 606 adhere to any of a number of accepted application protocols employed to communicate information between host applications and smart card applications. Accordingly, they need not be further described.

The DPDU 704 adheres to any of a number of accepted transport protocols used to communicate information between a smart card and a host system (e.g., a computer system). Fig. 6 breaks out DPDU 704 to denote a number of constituent fields including, but not limited to, a node address field 608, a protocol control block 610, a length field 612, a data field 614 and an error detection field 616. It is to be appreciated that the illustrated size of the fields do not necessarily correspond to relative block sizes, and the order of the fields may well be changed without deviating from the spirit and scope of the present invention.

The node address field 608 contains source and/or destination address information. More specifically, the node address field 608 includes information regarding the virtual address of the application issuing the frame (source address)

and the address of the application to which the frame is being sent (destination address). According to one aspect of the invention, a debug frame is denoted as such by embedding invalid source and/or destination addresses in the node address field 608. Upon receiving a communication flow, debug filter (304, 504) identifies the invalid source and/or destination address and routes the debug frame to a debug application or monitor.

The protocol control block 610 denotes whether the frame is a command frame or a response frame, and whether the received frame is the last frame in a communication, or whether more frames follow to deliver the data required to complete the communication instance.

As their names imply, the length field 612 provides an indication as to the length of the frame, while the data field 614 carries the instructions/data communicated between the host and the smart card. According to one implementation, the data field 614 includes debug instructions that direct a debug monitor 506 of a SCDI 110 to perform some task. Examples of such debug instructions (in pseudo code) and their effect include:

Debug (run app_x, Data) - execute an application using Data

Debug (get context) - obtain the context of smart card control logic

Debug (read memory) - read smart card memory

Debug (step) - step an application executing on the smart card.

Debug (set breakpoint) - set a breakpoint within an application executing on the smart card.

Debug (run) - execute application on smart card until event (e.g., breakpoint)

Additional debug instructions and their function can be found within the Appendix attached hereto.

The error checking field 616 of DPDU 604 includes information utilized by a development interface (i.e., CDI 114 or SCDI 110) to verify the integrity of the received frame. Any of a number of suitable error checking schemes may well be used such as, for example, inclusion of a checksum.

Example Operation

Fig. 7 is a flow chart of an example method for debugging an IC card application using an interlaced debug protocol, according to one aspect of the present invention. For ease of explanation, and not limitation, the method of Fig. 7 will be developed with continued reference to Fig.'s 1-6.

Turning to Fig. 7, the method begins with step 702 wherein the debug environment of a host computer 102 is invoked. In one implementation, the debug environment resides within application development tool 118, while in alternate embodiments, the debug environment is a stand-alone application 116.

In step 704, a debug command frame is generated within the debug environment. More specifically, a user instructs the debug environment to invoke a debug feature of coupled smart card 104. As described above, the debug environment marks the debug frame as such utilizing an invalid address in node address field 608 of the generated debug frame (e.g., DPDU 604). Once generated, the debug frame is sent to the CDI 114.

The CDI 114 receives the debug frame and interlaces the received debug frame with other application frames comprising the normal communication flow,

step 706. As described above, debug filter 304 identifies the received debug frame by detecting an invalid address within the node address field 608 of the received DPDU 604. Having interlaced the debug frame within the normal communication flow, CDI 114 transmits the communication flow to the smart card 104, via card reader 106 and communication medium 108.

In step 710, SCDI 110 receives the communication flow, and identifies the debug frame(s) interlaced within the communication flow in step 712. More specifically, SCDI 110 receives the communication flow via I/O interface 402, whereupon debug filter 504 detects one or more frames with an invalid address populating the node address field 608, while the error detection field 616 does not indicate any error of transmission. Accordingly, controller 502 concludes that such received frames are debug frames.

In response to receiving the communication flow with interlaced debug frames, steps 710 and 712, controller 502 of SCDI 110 promotes application frames to an appropriate smart card application 404, while the debug frames are routed to debug monitor 506, step 714. Controller 502 identifies the appropriate smart card application 404 to route the application frames using information (e.g., source and/or destination address information) embedded within node address field 608.

In response to receiving the communication frames, the appropriate smart card application 404 (i.e., the one to which the application frames were addressed) performs in accordance with the program code of the application and any instructions received in the application frames, subject to the debug monitor 506. Similarly, debug monitor 506 controls smart card resources according to debug instructions received in the debug frames to control execution of smart card

application 404 and/or to interrogate smart card resources. In response to execution of received communication frames (i.e., application and debug) within their respective applications (i.e., smart card application and debug monitor), smart card application 404 may generate response application frames. Similarly, debug monitor 506 may generate response debug frames depending, of course, on the point at which execution of the applications is suspended, step 716. In this manner, debug monitor 506 of SCDI 110, in response to the innovative interlaced debug protocol, manage execution of smart card applications, and disclosure of smart card state information to facilitate application development.

In step 718, SCDI 110 receives the response application frames generated by smart card application 404 and interlaces received response debug frames generated by debug monitor 506, if any, to generate a response communication flow. The communication flow is transmitted to the computer system 102 via communication medium 108 and card reader 106, step 720.

In step 722, CDI 114 receives the response communication flow from the smart card 104. More specifically, debug filter 304 receives and analyzes the response frames to detect debug frames. In step 724, controller 302 of CDI 114 promotes application frames to the host application 116, while identified debug frames are promoted to the debug environment.

Turning briefly to **Fig. 8**, a signaling diagram for an example communication session between a host system and a smart card utilizing the interlaced debug protocol of the present invention is presented. As shown, the communication is broken down according to the functional elements discussed above, namely, between application development tool 118, a debug environment 802, CDI 114, SCDI 110, debug monitor 506 and a smart card application 404.

According to one embodiment of the present invention, application frames are denoted by pseudo-code having the "Normal" tag, while debug frames are denoted by pseudo-code having a "Debug" tag.

The example communication session depicted in Fig. 8 adheres to the basic flow illustrated in Fig. 7 and, thus, will not be described in detail. Rather, the signaling diagram of Fig. 8 is intended to provide an example of how debug frames are interlaced within the normal communication flow between a host system and a smart card, and how the CDI 114 and the SCDI 110 promote the received frames accordingly.

Given the foregoing, it is to be appreciated that the innovative smart card development interface 110 and the interlaced debug protocol of the present invention transform the otherwise closed architecture of an otherwise typical smart card 104 into an application development tool. Moreover, the client development interface 114 transforms a common application development tool such as Microsoft's Visual BASIC, or Visual C/C++ into a smart card application development tool. Accordingly, the combination of the smart card development interface 110, the client development interface 114 and the interlaced debug protocol enable a developer to enter the smart card development market with minimal cost, thereby facilitating the development of applications for limited-sized markets and promoting the growth of the smart card industry.Although the invention has been described in language specific to structural features and/or methodological steps, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or steps described. Rather, the specific features and steps are disclosed as preferred forms of implementing the claimed invention.

CLAIMS

1. An integrated circuit (IC) card comprising:

an input/output (I/O) interface; and

a smart card development interface, coupled to the I/O interface, to receive and identify one or more debug frames interlaced within a normal communication flow between the IC card and a host system.

2. An IC card according to claim 1, further comprising:

a memory device having stored therein a plurality of executable instructions; and

a controller, coupled to the memory device and the smart card development interface, to execute at least a subset of the plurality of executable instruction to selectively implement one or more of a plurality of IC card applets.

3. An IC card according to claim 2, wherein the memory device includes a plurality of executable instructions which, when executed, implement a debug application which selectively controls other applications executing on the IC card.

4. An IC card according to claim 1, wherein the smart card development interface includes a debug filter to identify and remove the debug frames from the normal communication flow.

5. An IC card according to claim 4, wherein the debug filter redirects the debug frames to a debug application on the IC card.

6. An IC card according to claim 1, further comprising a debug application, responsive to debug instructions embedded within received debug frames, the debug application providing a user with a host of application debug features enabled in response to the received debug instructions.

7. An IC card according to claim 6, wherein select debug instructions invoke one or more of the following debug features: read/write IC card memory, get/set breakpoints in an IC card applet, sequentially step an IC card application, run an IC card applet, and release an IC card applet frame.

8. An IC card according to claim 1, wherein the IC card communicates with a remote host system using a transport protocol comprising application data units (APDU) and debug protocol data units (DPDU).

9. An IC card according to claim 8, wherein the transport protocol is a standard smart card communication protocol, wherein the APDU and the DPDU adhere to the standard IC card communication protocol.

10. An IC card according to claim 8, wherein the smart card development interface further comprises a debug filter which identifies DPDU within the normal communication flow to redirect the DPDU to a debug application on the IC card.

11. An IC card according to claim 10, wherein the debug filter identifies DPDU within the normal communication flow by detecting an invalid source and/or destination address identifier within the debug frame.

12. An IC card according to claim 1, wherein the IC card communicates with a remote host system using a transport protocol comprising application data units (APDU) including normal application frames and debug frames.

13. A storage medium having stored thereon a plurality of executable instructions which, when executed, implement the smart card development interface of claim 1.

14. A method of debugging a smart card application, the method comprising:

receiving one or more debug frames interlaced with application frames comprising a normal communication flow between a smart card and a host system;

identifying the one or more debug frames;

routing the received debug frames to a debug application executing on the smart card, while promoting the application frames to an application executing on the smart card, subject to conditions imposed by the debug frames.

15. A method according to claim 14, wherein the step of identifying the one or more debug frames comprises:

reading a source and/or destination address of frames comprising the normal communication flow; and

detecting invalid source and/or destination addresses in select frames denoting debug frames.

16. A method according to claim 14, further comprising:

implementing one or more debug features on the smart card according to debug instructions embedded within the received debug frames.

17. A method according to claim 16, wherein the debug features include one or more of read/write smart card memory, get/set breakpoints in a smart card application, sequentially step a smart card application, run a smart card application, and release a smart card application frame.

18. A method according to claim 14, further comprising:

generating a response debug frame to a received debug frame;

interlacing the response debug frame with response application frames; and

sending the response debug frame and response application frames to a host system.

19. A storage medium having stored thereon a plurality of instructions which, when executed, implement the method of claim 14.

20. A computer system comprising:

an input/output (I/O) interface; and

a client development interface, coupled to the I/O interface, to receive and identify debug frames interlaced within the normal communication flow between the computer system and a removably coupled smart card.

21. A computer system according to claim 20, further comprising:

a memory device having stored therein a plurality of instructions; and

a processor, coupled to the memory device and the client development interface, to execute at least a subset of the plurality of instructions to implement one or more applications including a smart card development application having a debug environment to send and receive debug frames to the coupled IC card interlaced within the normal communication flow between the computer system and the IC card.

22. A computer system according to claim 14, wherein the memory device includes a plurality of executable instructions which, when executed, implement a debug application on the computer system to communicate with and control smart card resources.

23. A computer system according to claim 20, wherein the client development interface includes a debug filter to identify and remove the debug frames from the normal communication flow between the computer system and the smart card.

24. A computer system according to claim 23, wherein the debug filter redirects debug frames received from the smart card to a debug application executing on the computer system.

25. A computer system according to claim 20, further comprising a debug application, to write and read debug frames to and from the smart card, facilitating a number of application debugging features.

26. A computer system according to claim 25, wherein the debug frames written by the debug application invoke one or more of the following debug features: read/write smart card memory, get/set breakpoints in a smart card application, sequentially step a smart card application, run a smart card application, and release a smart card application frame.

27. A computer system according to claim 20, wherein the computer system communicates with the smart card using a transport protocol comprising application data units (APDU) and debug protocol data units (DPDU).

28. A computer system according to claim 27, wherein the client development interface further comprises a debug filter which identifies DPDU within the normal communication flow to redirect the DPDU to a debug application executing on the computer system.

29. A computer system according to claim 29, wherein the debug filter identifies DPDU within the normal communication flow by detecting an invalid source and/or destination address identifier within the debug frame.

30. A storage medium having stored thereon a plurality of executable instructions which, when executed, implement the client development interface of claim 20.

31. A computer-implemented method for debugging a smart card application, the method comprising:

generating one or more debug frames containing debug instructions;

interlacing the generated debug frames with one or more application frames generated according to an application executing on the computer; and

sending the application frames with the interlaced debug frames to a removably coupled smart card, wherein the debug frames invoke one or more debug features of the smart card.

32. A computer-implemented method according to claim 31, wherein the application frames are generated by an application executing within an application development environment, while the debug frames are generated in response to user interaction with the smart card application development environment.

33. A computer-implemented method according to claim 31, wherein generating one or more debug frames comprises populating a source and/or destination field of the debug frame with an invalid source and/or destination address.

34. A computer-implemented method according to claim 31, further comprising:

receiving a normal communication flow from the smart card including debug frames interlaced with application frames, wherein the debug frames received from the smart card are received in response to debug frames issued by the computer.

35. A computer-implemented method according to claim 34, wherein the application frames are promoted to an associated application executing within an application development tool executing on the computer, while the debug frames are promoted to an application debug environment of the application development tool executing on the computer.

36. A communication protocol, employed between a host system and a smart card, the protocol comprising:

a plurality of application frames comprising a normal communication flow between a host application and a smart card application; and

one or more debug frames, interlaced with the application frames within the normal communication flow, to enable a debug application executing on the host system to selectively access and control smart card resources.

37. A communication protocol according to claim 36, wherein the debug application and the host application are executing on separate host systems, each communicatively coupled to the smart card.

38. A communication protocol according to claim 36, wherein the one or more debug frames include debug instruction to implement one or more of the following debug features: read/write smart card memory, get/set breakpoints in a smart card application, sequentially step a smart card application, run a smart card application, and release a smart card application frame.

39. A communication protocol according to claim 36, wherein the debug frame is distinguished from an application frame by incorporating an invalid source address.

40. A communication protocol according to claim 36, wherein the debug frame is distinguished from an application frame by incorporating an invalid destination address.

41. An application development system comprising:

a computer system to execute an application within an application development tool; and

a smart card incorporating a smart card development interface, coupled to the computer system, to receive and identify debug frames interlaced with application frames within a normal communication flow between the application executing on the computer system and the smart card, wherein the smart card development interface promotes the application frames to an application layer of the smart card, and invokes debug features of the smart card in response to debug instructions embedded within the received debug frames.

42. An application development system according to claim 41, wherein the computer system further comprises:

a client development interface, to interlace debug frames generated by the application development tool with application frames generated by the application executing within the application development tool.

43. An application development system according to claim 42, wherein the application development tool generates debug frames in response to user interaction with the application development tool.

44. An application development system according to claim 43, wherein the application development tool populates a source and/or destination field of the debug frame with an invalid source and/or destination address.

45. An application development system according to claim 43, wherein the debug frames invoke and control one or more smart card resources facilitating debugging of the application executing within the application development tool of the computer system.

46. An application development system according to claim 42, wherein the client development interface includes a debug filter to identify and route debug frames received from the smart card.

47. An application development system according to claim 41, wherein the smart card development interface comprises a debug filter to identify debug frames within the received normal communication flow.

48. An application development system according to claim 47, wherein the debug filter identifies debug frames by an invalid source and/or destination address embedded within a source and/or destination field of the debug frame.

49. An application development system according to claim 41, further comprising:

a communication protocol, employed by the computer system and the smart card to communicate therebetween, the communication protocol comprising,

a plurality of application frames comprising a normal communication flow between a host application and a smart card application; and

one or more debug frames, interlaced with the application frames within the normal communication flow, to enable a debug application executing on the host system to selectively access and control smart card resources.

ABSTRACT

An integrated circuit (IC) card is presented comprising an input/output (I/O) interface and a smart card development interface (SCDI), coupled to the I/O interface, to receive and identify debug frames interlaced within a normal communication flow between the IC card and a host system.

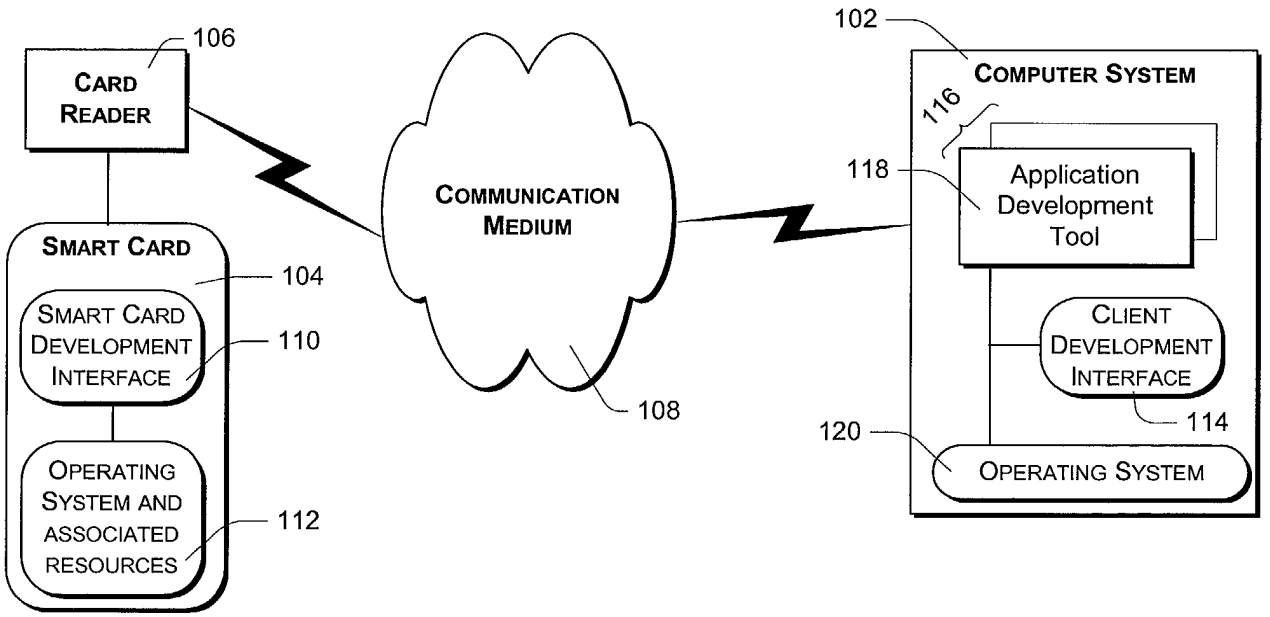

100
106
CARD READER
SMART CARD
104
SMART CARD DEVELOPMENT INTERFACE
110
OPERATING SYSTEM AND ASSOCIATED RESOURCES
112
COMMUNICATION MEDIUM
108
102
COMPUTER SYSTEM
116
118
Application Development Tool
CLIENT DEVELOPMENT INTERFACE
114
120
OPERATING SYSTEM

Fig. 1

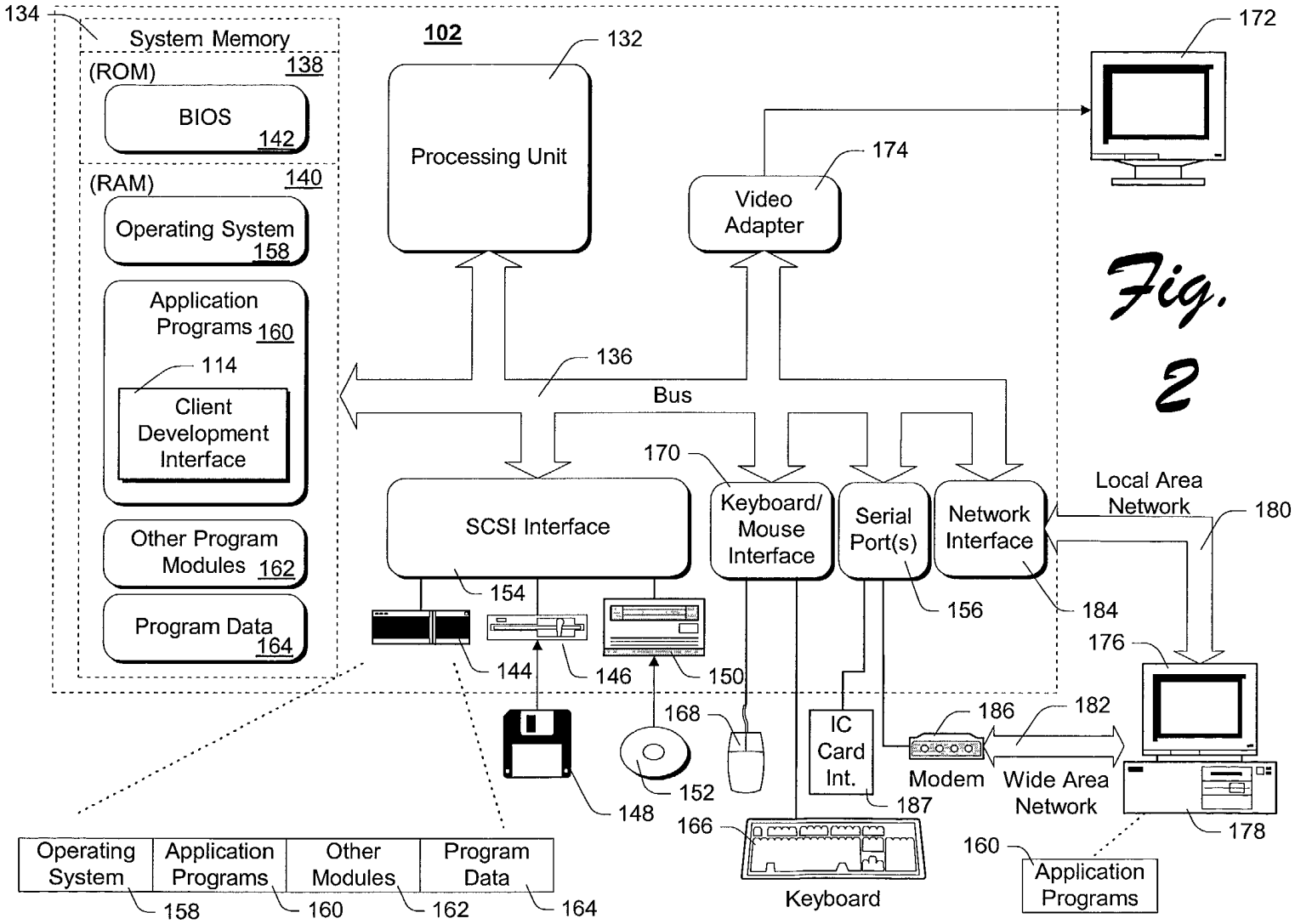


Fig. 2

MS1-385US

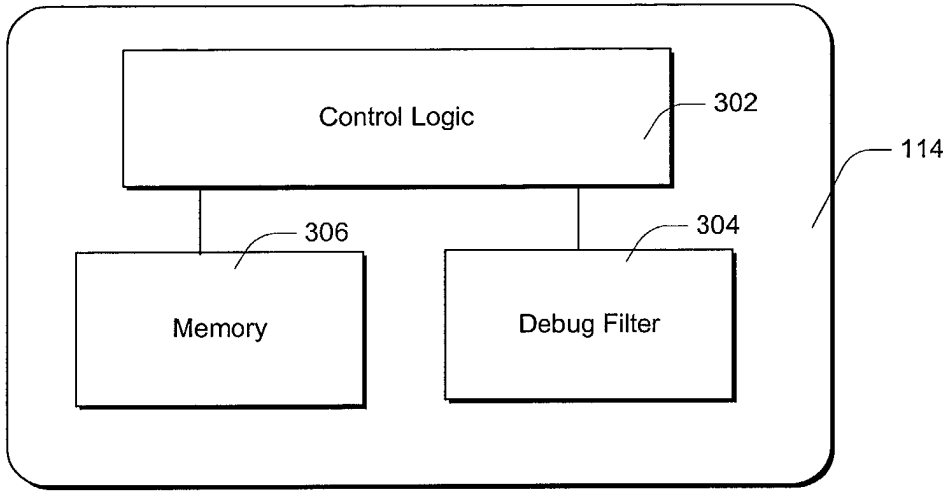

Control Logic
302
114
306
304
Memory
Debug Filter

Fig. 3

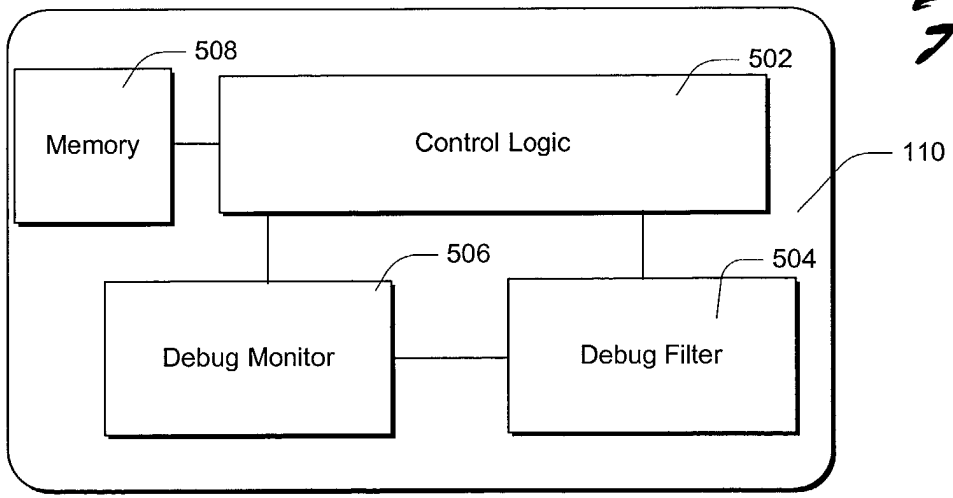

508
502
Memory
Control Logic
110
506
504
Debug Monitor
Debug Filter

Fig. 5

Fig. 4

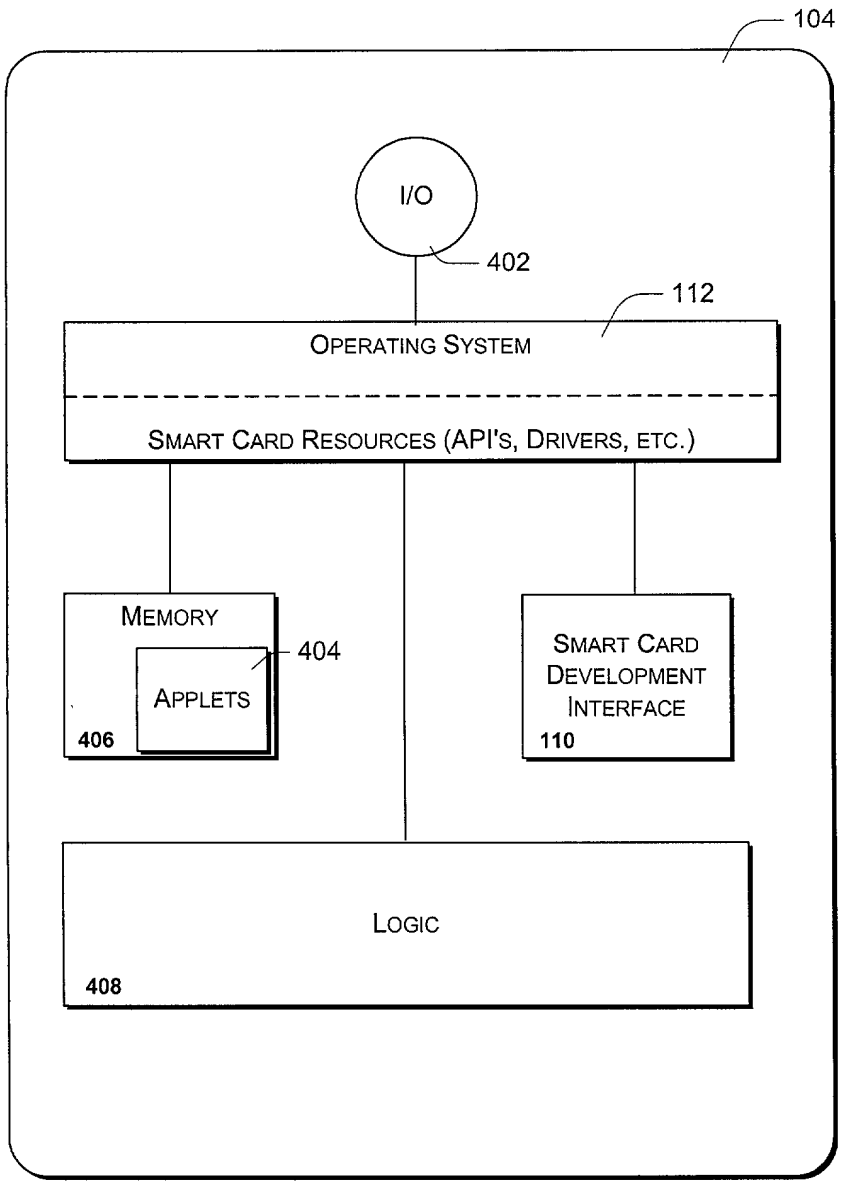

104
I/O
402
112
OPERATING SYSTEM
SMART CARD RESOURCES (API'S, DRIVERS, ETC.)
MEMORY
404
APPLETS
406
SMART CARD DEVELOPMENT INTERFACE
110
LOGIC
408

Fig. 6

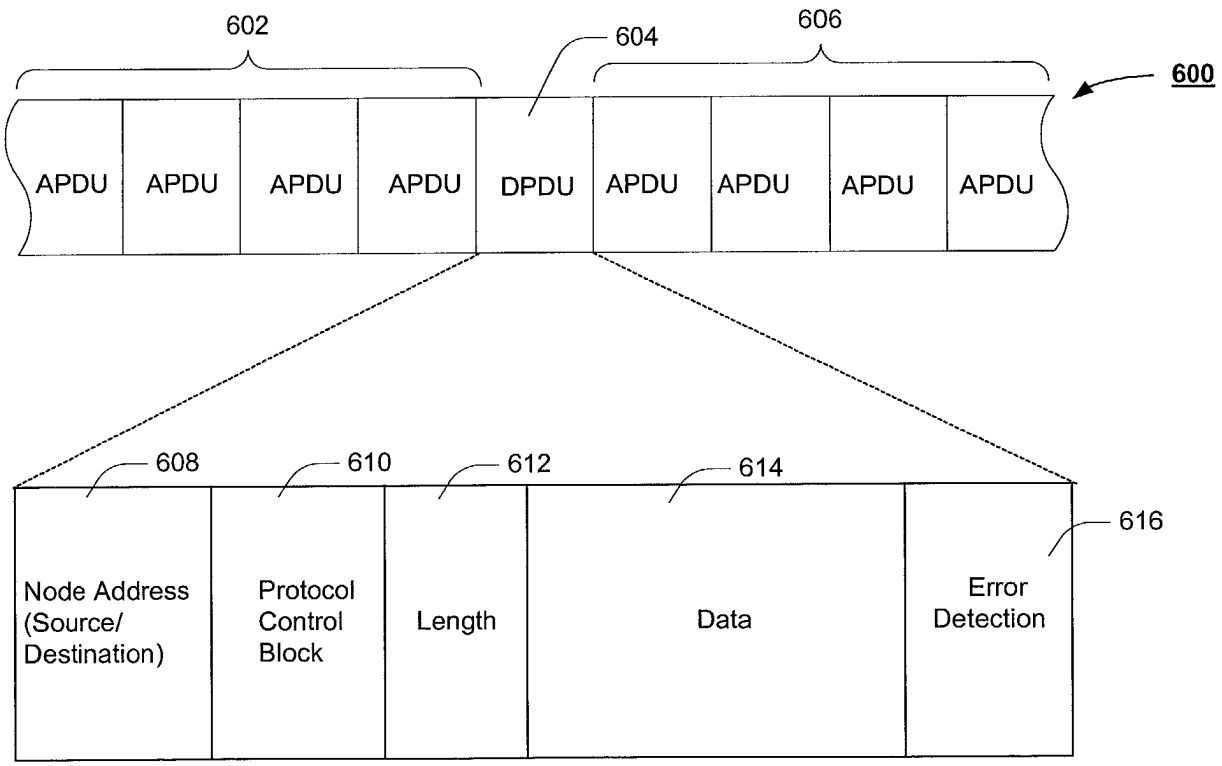

602
604
606
600
APDU
APDU
APDU
APDU
DPDU
APDU
APDU
APDU
APDU
608
610
612
614
616
Node Address (Source/ Destination)
Protocol Control Block
Length
Data
Error Detection

Fig 7

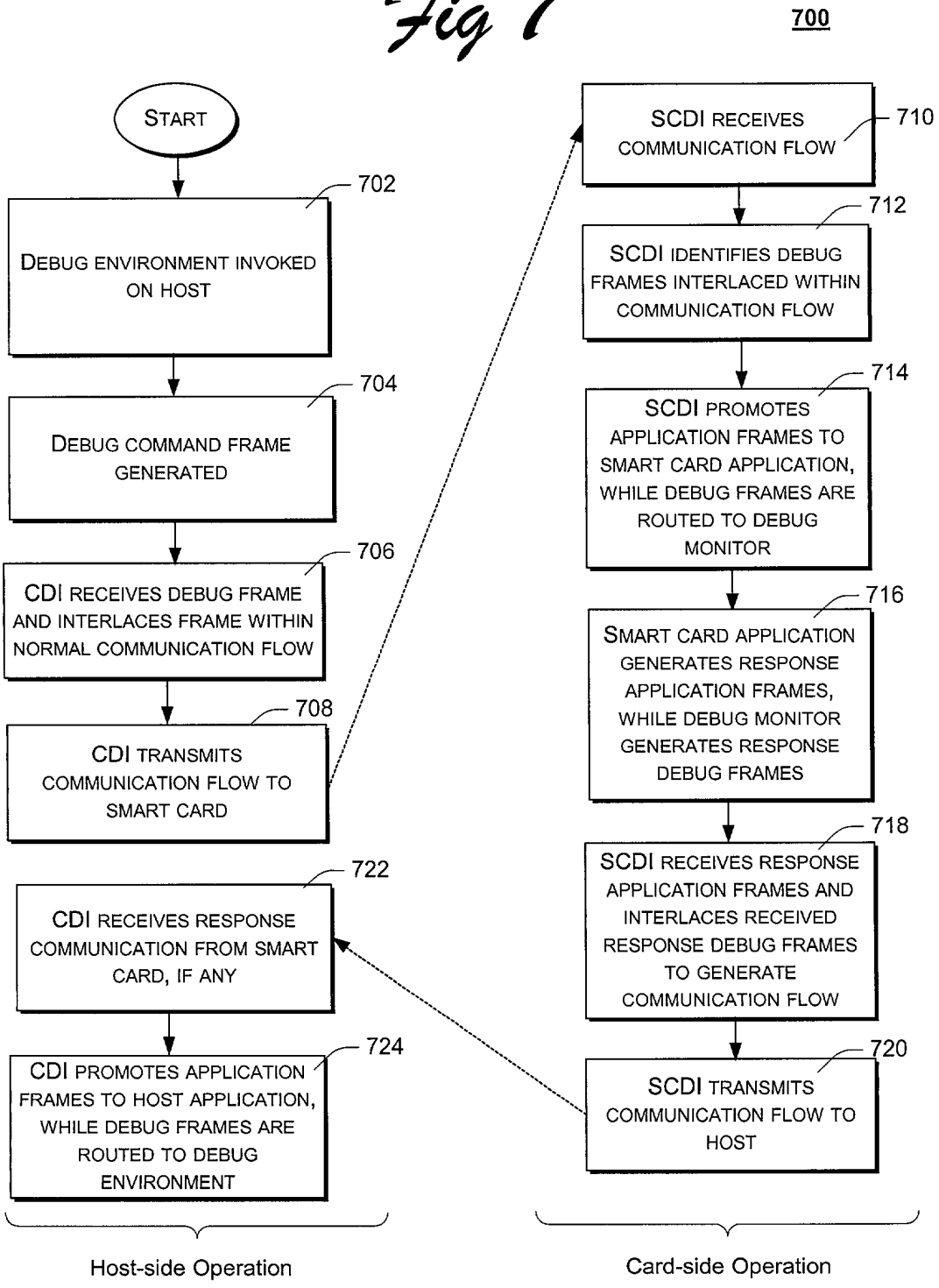
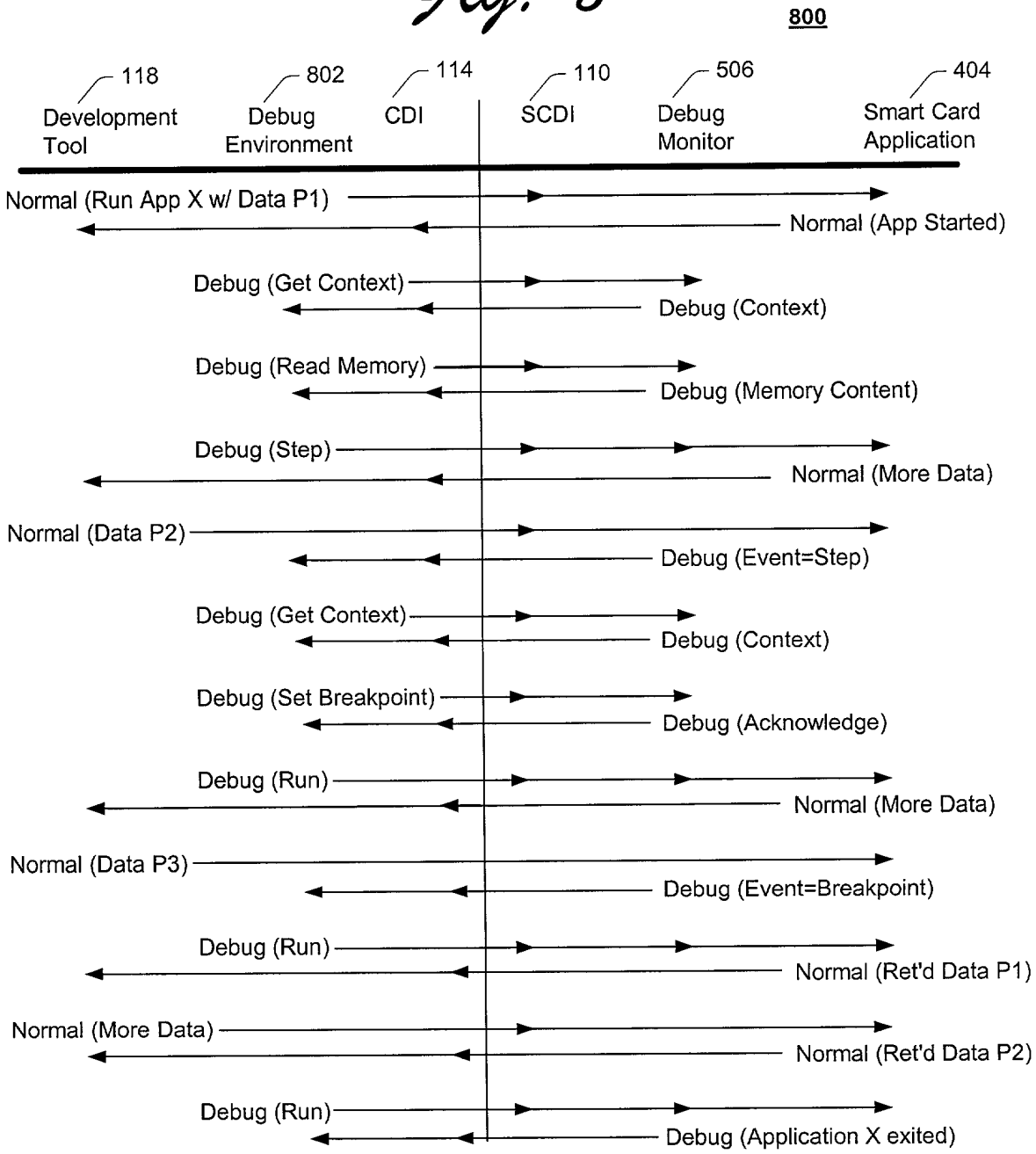

700
START
702
DEBUG ENVIRONMENT INVOKED ON HOST
704
DEBUG COMMAND FRAME GENERATED
706
CDI RECEIVES DEBUG FRAME AND INTERLACES FRAME WITHIN NORMAL COMMUNICATION FLOW
708
CDI TRANSMITS COMMUNICATION FLOW TO SMART CARD
722
CDI RECEIVES RESPONSE COMMUNICATION FROM SMART CARD, IF ANY
724
CDI PROMOTES APPLICATION FRAMES TO HOST APPLICATION, WHILE DEBUG FRAMES ARE ROUTED TO DEBUG ENVIRONMENT
Host-side Operation
SCDI RECEIVES COMMUNICATION FLOW
710
712
SCDI IDENTIFIES DEBUG FRAMES INTERLACED WITHIN COMMUNICATION FLOW
714
SCDI PROMOTES APPLICATION FRAMES TO SMART CARD APPLICATION, WHILE DEBUG FRAMES ARE ROUTED TO DEBUG MONITOR
716
SMART CARD APPLICATION GENERATES RESPONSE APPLICATION FRAMES, WHILE DEBUG MONITOR GENERATES RESPONSE DEBUG FRAMES
718
SCDI RECEIVES RESPONSE APPLICATION FRAMES AND INTERLACES RECEIVED RESPONSE DEBUG FRAMES TO GENERATE COMMUNICATION FLOW
720
SCDI TRANSMITS COMMUNICATION FLOW TO HOST
Card-side Operation

Fig. 8


800
118
Development Tool
802
Debug Environment
114
CDI
110
SCDI
506
Debug Monitor
404
Smart Card Application
Normal (Run App X w/ Data P1)
Normal (App Started)
Debug (Get Context)
Debug (Context)
Debug (Read Memory)
Debug (Memory Content)
Debug (Step)
Normal (More Data)
Normal (Data P2)
Debug (Event=Step)
Debug (Get Context)
Debug (Context)
Debug (Set Breakpoint)
Debug (Acknowledge)
Debug (Run)
Normal (More Data)
Normal (Data P3)
Debug (Event=Breakpoint)
Debug (Run)
Normal (Ret'd Data P1)
Normal (More Data)
Normal (Ret'd Data P2)
Debug (Run)
Debug (Application X exited)

APPENDIX A: DEBUG PROTOCOL ELEMENTS

- **Protocol details:**

Example of coding of I-blocks (ISO 7816-3 amendment 2) (bits b8 (most significant bit) to b1 (least significant bit)):

b8 = 0 denotes a I-block

b7 is the block number

b6 is the More-data bit (0 indicates last block of sequence)

b5-b1 not used

According to one implementation, command or response is implied by the direction of the exchange (i.e., host sends commands, card send responses).

- **Debug functions:**

GetContext: Retrieve current IC Card context information.

SetContext: Set current IC Card context.

Go (or Run): Execute an application or applet.

RunToPC: Execute an application or applet until program counter (PC) is reached; akin to a step function.

WriteMemory: Write to a specified location in IC Card Memory

ReadMemory: Read a specified location in IC Card Memory

GetNextBranch: Retrieve the next branch.

- **Debug Events:**

AppStart: denotes application/applet starts.

AppExit: denotes application/applet exits.

BreakPoint: Identify a PC wherein execution of application/applet is halted.

Step: Incrementally execute application/applet

Exception: denotes an exception.

DebugString: a string of information sent in a debug frame to host.

SimulatorExit: denotes the end of execution of simulator.